wCorr Arguments

Paul Bailey, Ahmad Emad, Ting Zhang, Qingshu Xie

2016-04-06

The wCorr package can be used to calculate unweighted or weighted correlations of the Pearson, Spearman, polyserial, and polychoric types. By extension, the package also impelments the tetrachoric correlation as a specific case of the polychoric correlation and biserial as a specific case of the polyserial correlation. When weights are used the correlation coefficients are calculated with so called sample weights or inverse probability weights.[1]

This vignette describes the implications of two Boolean switches in the wCorr package. First, the `ML` switch allows for either a non-MLE (but consistent) esitimate of the nuisance parameters that define the binning process to be used (`ML=FALSE`) or for the nuisance parameters to be estimated using the MLE (`ML=TRUE`). Second the `fast` argument gives the option to use a pure R implementation (`fast=FALSE`) or an implementation that relies on the `Rcpp` and `RcppArmadillo` packages (`fast=TRUE`).

The *wCorr Formulas* vignette describes the statistical properties of the correlation estimators in the package and has a more complete derivation of the likelihood functions.

The `ML` switch

The wCorr package computes correlation coefficients between two vectors of random variables that are jointly bivariate normal. We call the two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$.

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N\left[\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \boldsymbol{\Sigma}\right]$$

where $N(\mathbf{A}, \boldsymbol{\Sigma})$ is the bivariate normal distribution with mean $\boldsymbol{A}$ and covariance $\boldsymbol{\Sigma}$.

Computation of polyserial correlation

The likelihood function for an individual observation of the polyserial correlation is[2]

$$\Pr\left(\rho = r, \boldsymbol{\theta}; Z = z_i, M = m_i\right) = \phi(z_i)\left[\Phi\left(\frac{\theta_{m_i+2} - r \cdot z_i}{\sqrt{1-r^2}}\right) - \Phi\left(\frac{\theta_{m_i+1} - r \cdot z_i}{\sqrt{1-r^2}}\right)\right]$$

where ρ is the correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$, $\boldsymbol{Z}$ is the normalized version of $\boldsymbol{X}$, and $\boldsymbol{M}$ is a discretized version of $\boldsymbol{Y}$, using $\boldsymbol{\theta}$ as cut points as described in the *wCorr Formulas* vignette.

The log-likelihood is then

$$\ell(\rho, \boldsymbol{\theta}; z, m) = \sum_i w_i \ln\left[\Pr\left(\rho = r, \boldsymbol{\theta}; Z = z_i, M = m_i\right)\right]$$

The derivatives of ℓ can be written down but are not readily computed. When the `ML` argumet is set to `FALSE` (the default), a one dimensional optimization of ρ is calculated using `stats::optimize`. When the `ML` argument is set to `TRUE`, a multi-dimensional optimization is done for ρ and $\boldsymbol{\theta}$ using `minqa::bobyqa`.

[1] Sample weights are comperable to `pweight` in Stata.

[2] See the *wCorr Formulas* vignette for a more complete description and motivation for the polyserial correlations's likelihood function.

Computation of polychoric correlation

For the polychoric the observed data is discreteized for both variables. Here the discretized version of $\boldsymbol{X}$ is $\boldsymbol{P}$ and the discretized version of $\boldsymbol{Y}$ remains $\boldsymbol{M}$.[3] The likelihood function for the polychoric is

$$\Pr\left(\rho = r, \boldsymbol{\theta}, \boldsymbol{\theta}'; P = p_i, M = m_i\right) = \int_{\theta'_{p_i+1}}^{\theta'_{p_i+2}} \int_{\theta_{m_i+1}}^{\theta_{m_i+2}} f(x, y|\rho = r) dy dx$$

where $f(x, y|r)$ is the noramlzied bivariate normal distribution with correlation ρ, $\boldsymbol{\theta}$ are the cut points used to discretize Y into M, and $\boldsymbol{\theta'}$ are the cut points used to discretize $\boldsymbol{X}$ into $\boldsymbol{P}$.

The log-likelihood is then

$$\ell(\rho, \boldsymbol{\theta}, \boldsymbol{\theta}'; \mathbf{p}, \mathbf{m}) = \sum_i w_i \ln\left[\Pr\left(\rho = r, \boldsymbol{\theta}, \boldsymbol{\theta}'; P = p_i, M = m_i\right)\right]$$

The derivatives of ℓ can be written down but are not readily computed. When the `ML` argumet is set to `FALSE` (the default), a one dimensional optimization of ρ is calculated using `stats::optimize`. When the `ML` argument is set to `TRUE`, a multi-dimensional optimization is done for ρ, $\boldsymbol{\theta}$, and $\boldsymbol{\theta'}$ using `minqa::bobyqa`.

General procedures of the simulation study of unweighted correlations

A simulation is run several times. For each itteration, the following procedure is used:

- select a true correlation coefficient ρ;
- select the number of observations (n);
- generate $\boldsymbol{X}$ and $\boldsymbol{Y}$ to be bivariate normally distributed using a pseudo-Random Number Generator (RNG);
- using a pseudo-RNG, select the the number of bins for $\boldsymbol{M}$ and $\boldsymbol{P}$ (t and t') independantly from the set {2, 3, 4, 5};
- select the bin boundaries for $\boldsymbol{M}$ and $\boldsymbol{P}$ ($\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$) by sorting the results of $(t-1)$ and $(t'-1)$ draws, respectively, from a normal distribution using a pseudo-RNG;
- confirm that at least 2 levels of each of $\boldsymbol{M}$ and $\boldsymbol{P}$ are occupied (if not, retrun to generating $\boldsymbol{X}$ and $\boldsymbol{Y}$); and
- calculate and record relevant statistics.

When the exact method of selecting a parameter (such as n) is not noted above, it is described as part of each simulation.

ML switch

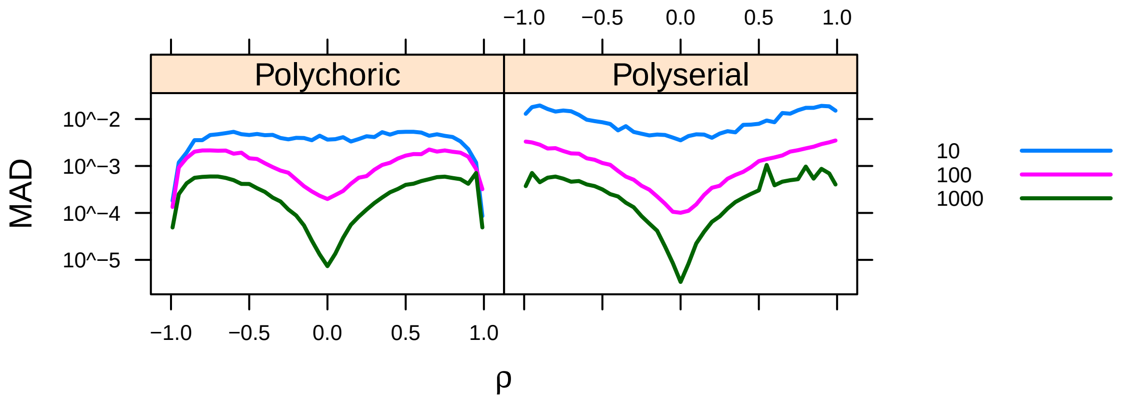
A simulation was done at each level of the cartesian product of $\texttt{ML} \in \{\texttt{TRUE}, \texttt{FALSE}\}$, $\rho \in (-0.99, -0.95, -0.90, -0.85, ..., 0.95, 0.99)$, and $n \in \{10, 100, 1000\}$. For precision, each iteration is run three times. The same values of the variables are used in the computation for `ML=TRUE` as well as for `ML=FALSE`; and then the statistics are compared between the two sets of results. Mean absolute difference (MAD) between the maximum likelihood estimator (`ML=TRUE`) and the estimator that uses a consistent estimate of $\boldsymbol{\theta}$ (and potentially $\boldsymbol{\theta'}$) and then maximizes over ρ only (`ML=FALSE`). It is given by

[3] See the "wCorr Formulas" vignette for a more complete description and motivation for the polychoric correlations's likelihood function.

$$MAD = |r_{ML=TRUE} - r_{ML=FALSE}|$$

where $r_{ML=TRUE}$ is the estimated correlation when `ML=TRUE`; and $r_{ML=FALSE}$ is the estimated correlation when `ML=FALSE`. Thus, this MAD is not deviation from the true ρ but the difference between the `ML` values.

This is a plot of the MAD as a function of the true correlation coefficnet. It shows a decrease in MAD with the increase of n increases (change from line to line), a decrease when the correlations are in the neighborhood of zero that is more pronounced for larger n, and a dip when the correlation is exactly 1 or -1.



This table shows the MAD by n and correlation type.

Correlation type	n	MAD
Polychoric	10	0.0038823
Polychoric	100	0.0012159
Polychoric	1000	0.0003116
Polyserial	10	0.0098884
Polyserial	100	0.0013830
Polyserial	1000	0.0003397

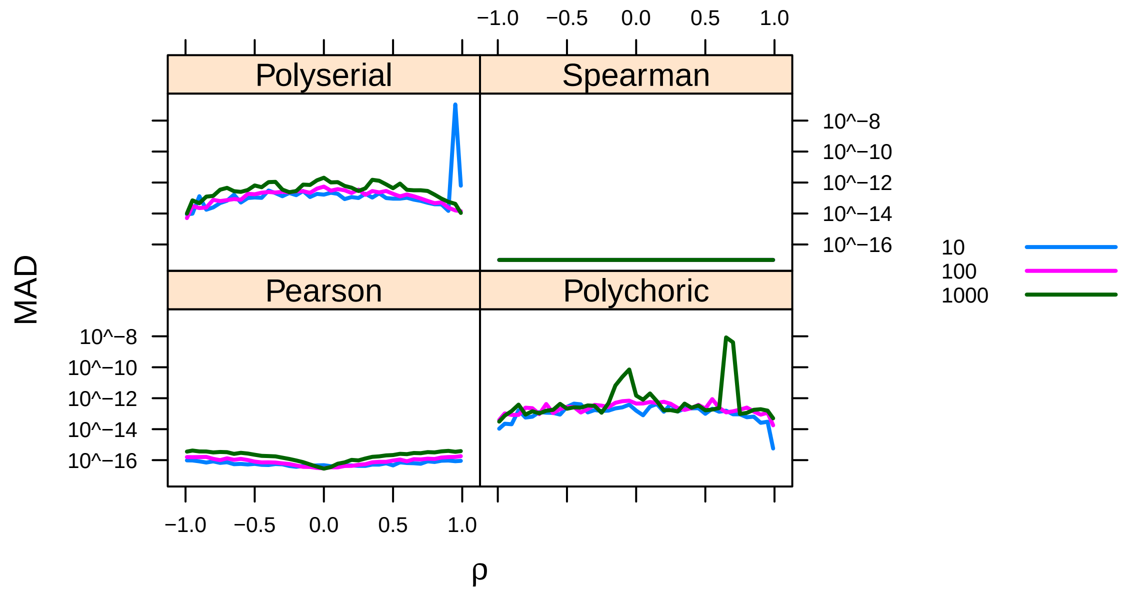
These resuls show that the size of the MAD decreases as n incrreases. When the sample size is $n = 10$, the MAD is less than 0.01 for the polyserial and 0.004 for the polychoric.

fast switch

This section examines the agreement between the pure R implementation of the optimizations and the `Rcpp` and `RcppArmadillo` impelemntation. The code can compute with either option by setting `fast=FALSE` (pure R) or `fast=TRUE` (Rcpp).

A simulation was done at each level of the cartesian product of `fast` $\in \{\texttt{TRUE}, \texttt{FALSE}\}$, $\rho \in (-0.99, -0.95, -0.90, -0.85, ..., 0.95, 0.99)$, and $n \in \{10, 100, 1000\}$. Each iteration was run 100 times. The same values of the variables are used in the computation for `fast=TRUE` as well as for `fast=FALSE`; and then the statistics are compared between the two sets of results.

The plot below shows all differences between the `fast=TRUE` and `fast=FALSE runs` for the four types of correlations. Note that differences smaller than 10^{-16} are indistinguishable from 0 by the machine. However, a factor of 10^{-17} was added to the results so that they could all be shown on a log scale.

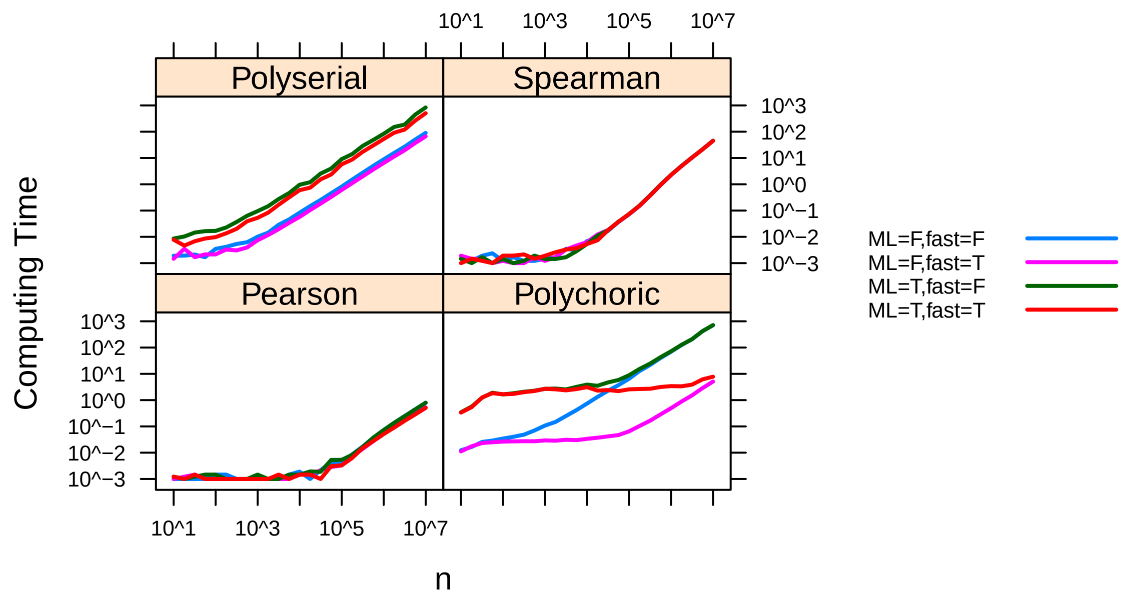


The above shows that differences as a result of the `fast` argument are never larger than 10^{-8} for any type. The Spearman never shows any difference that is different from zero and the Pearson show differences just larger than the smallest observable difference when using double precision floating point values (about 1×10^{-16}.

Implications for speed

A simulation was done at each level of the cartesian product of `ML` $\in$ {`TRUE`, `FALSE`}, `fast` $\in$ {`TRUE`, `FALSE`}, $\rho \in (-0.99, -0.95, -0.90, -0.85, ..., 0.95, 0.99)$, and $n \in \{10^1, 10^{0.75}, 10^{1.5}, ..., 10^7\}$. For precision, each iteration is run 80 times when $n < 10^5$ and 20 times when $n \geq 10^5$. The same values of the variables are used in the computations at all four combinations of `ML` and `fast`. A variety of correlations are chosen so that the results represent an average of possible values of ρ.

The following plot shows the mean computing time versus n.



Conclusion

Using tables presented in this vignette, users who wish to use the more accurate `ML=TRUE` argument can compare the difference in computing time and the difference in results.

The `fast` argument is provided primarily for comparison of the `Rcpp` and pure R code and shows agreement to within 10^{-8}.